Kevin Shih

Speech-2-Text Lock Documentation

Introduction:

This Python script was created to be implemented with a Raspberry Pi to serve as a speech activated device. Applications for this IoT device include door locks, lock boxes, lights, camera, text readouts, or any other output device. Furthermore, this system could utilize relays which increases the amount of devices that can be created and used alongside the system.

The script itself aims to allow people the ability to replace the need to use a button or switch to activate components and other electronics. This is achieved using the Google Speech API to perform the dictation from speech to text. Furthermore, a microphone needs to be plugged into the Raspberry Pi to allow it to record speech.

Implementation:

The program uses Google Speech API and pyaudio to record speech snippets, dictate them and print the results back to the user. If the results match the passkey hard coded by the user, the program then triggers the output devices such as LEDs, servo motors, solenoids, etc. Furthermore, a button is utilized with a while pressed function to only have the system listen for speech when the button is pressed. This allows the program to know when to stop listening and send the speech snippet to the Google Speech API instead of waiting to see if the user is still speaking. This improves accuracy, effectiveness, and speed.

Program Objectives:

The program should allow users to activate electronic devices and components with the security of a secret word or phrase. Moreover, the system must be reliable and operate smoothly with at least 80% accuracy. At the same time, having the shortest possible processing time will only make the user experience better.

User Interaction:

On the user side, the user must first change the “passkey” variable with a string the user wishes to use as the password or pass-phrase. After that as long as the script is running, the system is always ready for action. All the user needs to do is hold down the button, say the password or phrase, and release the button. After that the system will take care of the rest. If the password stated was incorrect, no action will be taken. If the system could not dictate the speech, it would issue a “Failed” statement. If the password is correct, the programmed actions will be executed. After all this, the system would once again be ready for the user to push the button and state the password again. Furthermore, the user can also say “quit” instead of a password to turn the system off.

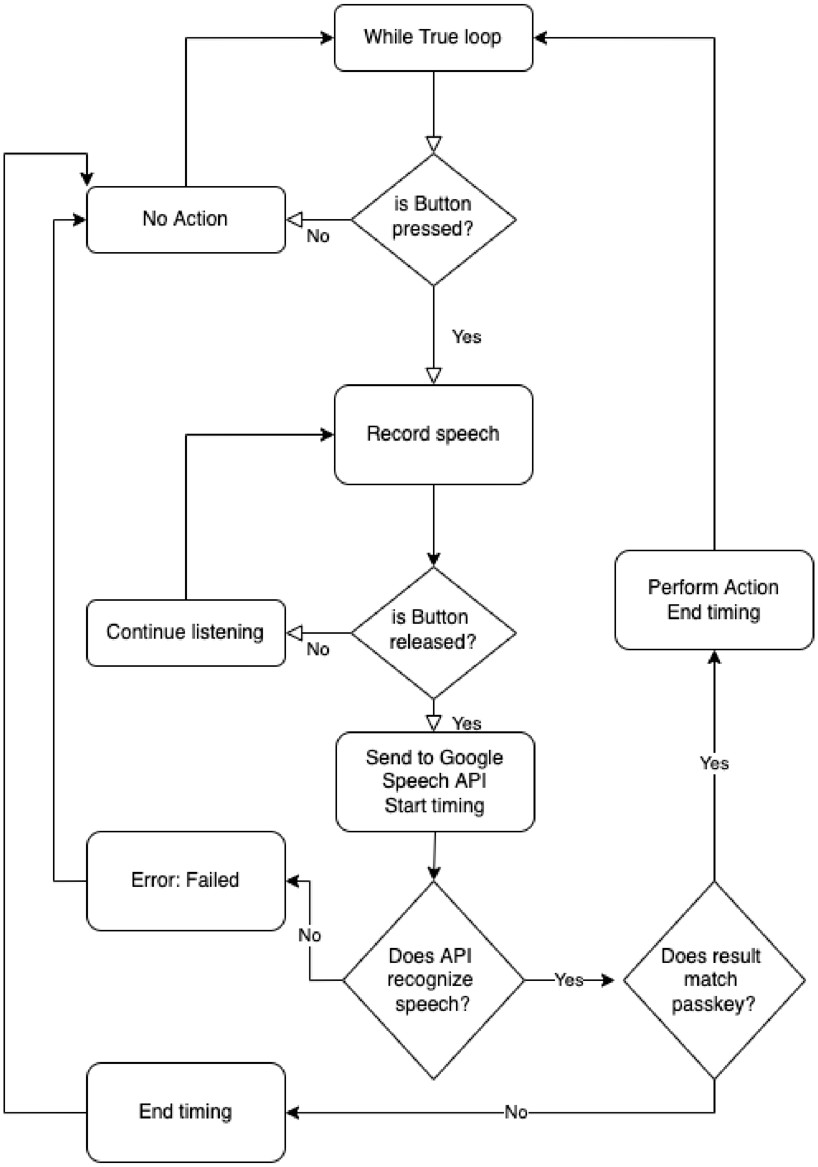
Constraints:

Due to its reliance on Google Speech API, the system must be connected to the internet and much of the processing time is spent waiting for the speech to be sent, dictated, and returned from the Google Speech API. Furthermore, unless relays are used, the maximum volts that can be supplied to output devices is 5 volts, making the system either very large or not easily portable.

Possible Fixes:

It is possible to use paid for Speech Recognition SDKs to get speech recognition functionality without the need for internet connectivity. This would greatly increase the speed and efficiency of the system. When it comes to power and portability of the system and its output devices, specialized portable power supplies capable of delivering 12 volts or more can be used.

Flowchart:


While True loop
is Button pressed?
No
No Action
Yes
Record speech
is Button released?
No
Continue listening
Yes
Send to Google Speech API Start timing
Does API recognize speech?
No
Error: Failed
Yes
Does result match passkey?
Yes
Perform Action End timing
No
End timing